arXiv:2602.18640v2 [cs.AI] 24 May 2026

Decoding ML Decision: An Agentic Reasoning Framework for Large-Scale Ranking System

Longfei Yun*, **Yihan Wu***, **Haoran Liu**†*, **Xiaoxuan Liu**, **Ziyun Xu**, **Yi Wang**, **Yang Xia**, **Pengfei Wang**, **Mingze Gao**, **Yunxiang Wang**, **Changfan Chen**, **Wenjie Fu**, **Hong Yan**, **Junfeng Pan**†

Meta

*Equal Contribution, {loyun,yihanwu,ryan0814}@meta.com

Modern large-scale ranking systems operate within a landscape of competing objectives, operational constraints, and evolving product requirements. Progress in this domain is increasingly bottlenecked not by modeling techniques alone, but by the *engineering context constraint*: the arduous process of translating ambiguous product intent into executable, verifiable hypotheses that account for feature stability, deployment feasibility, and multi-objective trade-offs. We present GEARS (Generative Engine for Agentic Ranking Systems), an orchestration framework that decomposes ranking optimization into three coordinated stages: (1) programmatic candidate generation and filtering, which translates natural-language intent into deterministic operations over tabular policy data; (2) domain-grounded interpretation via Specialized Agent Skills, which provide on-demand access to ranking-specific knowledge such as feature definitions and stability diagnostics; and (3) deterministic lifecycle governance, which enforces feature-stability and statistical-significance checks before any policy is promoted for deployment. We evaluate GEARS on a benchmark of 100 structured policy-selection instructions derived from 20 large-scale online experiments (0.5B users per experiment, 10B user-experiment pairs total), where it achieves 0.94 nDCG@1, outperforming prompting baselines and code-execution methods. Ablation analysis reveals that programmatic filtering is the primary driver on structured tasks, while specialized skills provide complementary gains on tasks requiring contextual interpretation. Deployment across nine production surfaces demonstrates consistent metric improvements with an annual LLM cost under $1,500 and a reduction in engineering cycle time from weeks to days.

Correspondence: Haoran Liu at ryan0814@meta.com, Junfeng Pan at panjunfeng@meta.com

Meta

1 Introduction

Modern ranking systems within large-scale social media platforms orchestrate heterogeneous product surfaces, ranging from discovery-oriented recommendation to community-driven social interfaces, serving a billion-scale global user base with multifaceted preferences. Over decades of iterative development, these systems have evolved into highly intricate architectures where numerous optimization layers concurrently target diverse and often conflicting metrics. Consequently, the primary bottleneck to system advancement has shifted from pure signal estimation to the *engineering context constraint*: the arduous translation of product intuition and domain expertise into auditable, executable hypotheses. Current industry workflows remain tethered to manual intervention, relying on domain experts to navigate the combinatorial complexity of multi-objective trade-offs, treatment interpretability, and the rigorous alignment with evolving business criteria. This manual dependence creates a scalability barrier, leaving high-value policies undiscovered within the search space.

Furthermore, traditional optimization approaches, such as uplift modeling (Künzel et al., 2019; Zhao et al., 2017; Wei et al., 2024), treat personalization as a static model selection task. While these methods may identify statistically promising interventions, they frequently fail to account for operational constraints or feature instability, resulting in policies that are optimal offline but brittle or undeployable in production environments. We refer to this disconnect between statistical optimality and production feasibility as the *deployment gap*.

To bridge this gap, we introduce GEARS (Generative Engine for Agentic Ranking Systems), a framework that decomposes the end-to-end ranking optimization workflow into three coordinated stages, orchestrated by an LLM-based controller. Rather than attempting to solve for optimality in a single inference step, GEARS treats the experimentation ecosystem as an

interactive environment where each stage applies the most appropriate tool for the sub-task at hand:

1. **Programmatic candidate generation.** The system translates a natural-language product intent into executable search specifications, generates candidate policies via tolerance-based Pareto filtering, and applies deterministic pre-filtering through code execution. This stage handles structured, data-manipulation operations where programmatic execution is more reliable than text-based reasoning.
2. **Domain-grounded interpretation.** Specialized Agent Skills provide the controller with on-demand access to ranking-specific knowledge—feature definitions, stability diagnostics, and trade-off explanations—that is absent from both the LLM's pretraining data and the tabular experiment records. Skills are loaded lazily to bound context length and mitigate context rot in long sessions.
3. **Deterministic lifecycle governance.** Validation hooks enforce production safety by checking feature stability and statistical significance before any policy is promoted. Only candidates that pass all deterministic checks are included in the final recommendation set.

This design reflects a deliberate division of labor: code execution handles what it does best (filtering, sorting, ranking over tabular data), domain skills supply contextual knowledge that code alone cannot provide, and governance hooks ensure deployability. The LLM-based controller orchestrates these heterogeneous components rather than attempting to replace them.

Experimental validation on a benchmark of 100 structured policy-selection instructions derived from 20 billion-user-scale online experiments demonstrates that GEARS achieves 0.94 nDCG@1, outperforming prompting baselines and code-execution methods. Ablation analysis confirms that programmatic filtering is the primary contributor on structured tasks, while specialized skills provide complementary gains on tasks requiring contextual interpretation. Deployment across nine production surfaces delivers consistent metric improvements, reducing the engineering cycle from weeks to days at an annual LLM cost under $1,500.

Our contributions are threefold:

1. **Orchestration framework for ranking optimization.** We decompose ranking policy selection into programmatic filtering, domain-grounded interpretation, and deterministic governance, coordinated by an LLM-based controller.
2. **Specialized Agent Skills.** We introduce modular, on-demand knowledge resources that externalize ranking expertise into reusable, auditable capabilities.
3. **Production validation at scale.** We demonstrate the effectiveness of GEARS across nine production surfaces serving billions of users, with quantitative analysis of each component's contribution.

2 Related Works

The Evolution of Context Engineering LLMs have evolved from simple instruction-following systems into core reasoning engines, necessitating a shift from *prompt engineering* to the formal discipline of *Context Engineering* (Mei et al., 2025; Amatriain, 2024; Ye et al., 2024; Velásquez-Henao et al., 2023). This paradigm shift reconceptualizes the input context not as a monolithic, static string, but as a dynamically structured assembly of informational components. The development of Tool-Integrated Reasoning (TIR) has further transformed LLMs from passive text generators into *world interactors* capable of autonomous environmental manipulation through structured function calling (Qian et al., 2025; Mialon et al., 2023; Dong et al., 2025; Chen et al., 2022; Li et al., 2023). Furthermore, contextual self-refinement mechanisms, such as Self-Refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023), demonstrate that models can improve their own output quality through iterative feedback loops and reflective episodic memory. To address the inherent statelessness of LLMs, research into Memory Systems has introduced OS-inspired hierarchical storage and cognitive architectures. Implementations such as MemGPT (Packer et al., 2023) leverage virtual memory paging to traverse limited context windows, while frameworks like MemoryBank (Zhong et al., 2023) utilize cognitive principles, such as the Ebbinghaus forgetting curve, to dynamically update memory strength. Within the domain of Multi-Agent Systems (MAS), current research prioritizes sophisticated communication protocols and orchestration mechanisms (Anthropic, 2024, 2025b,a).

Uplift Modeling and HTE Traditional uplift modeling approaches, which seek to identify user segments most responsive to specific interventions, can be broadly categorized into meta-learner based methods, tree-based methods, and neural network-based methods. Meta-learner based methods utilize existing prediction models to estimate individual treatment effects (ITE). Approaches such as S-learner and T-learner (Künzel et al., 2019) either combine treat-

ment variables with user features in a single model or build separate models for control and treatment groups, respectively. While effective, these paradigms can suffer from performance degradation when there is a significant imbalance in data between groups. Tree-based (Zhao et al., 2017; Radcliffe and Surry, 2011; Athey and Imbens, 2016; Nandy et al., 2023) methods employ decision trees or forests to partition the user population into subgroups based on their sensitivity to different treatments, using treatment information as part of the splitting criteria. These methods are valued for their interpretability and ability to uncover heterogeneous treatment effects. Neural network-based methods (Wei et al., 2024; Louizos et al., 2017; Bica et al., 2020; Sun and Chen, 2024; Shi et al., 2019) leverage the representational power of deep learning to model complex user responses and estimate uplift. By introducing flexible architectures, these methods can capture intricate relationships between user features and interventions, and have been widely adopted in domains such as online marketing and precision targeting. Meta has also been investing in this area over a decade, such as Smart Scorer Peysakhovich and Lada (2016); Lada et al. (2019),but those methods are only intended to solve algorithmic problems, without incorporating any ranking context.

Despite their strengths, most existing approaches rely heavily on offline features and static user profiles, which limits their ability to respond to real-time shifts in user behavior or to dynamically adapt intervention strategies. In practice, even when an algorithm delivers the "best" offline result, it may never ship, because large-scale ranking systems are highly complex. Any deployable solution must account for business objectives, existing ranking policies, and cannibalization across products and systems. Furthermore, the design and deployment of these models often require manual workflows that are time-consuming and resource-intensive.

Adaptive Experimentation (AE) Distinct from static prediction models, Adaptive Experimentation (AE) focuses on optimizing outcomes through sequential decision-making and active exploration. AX methodologies, including Multi-Armed Bandits and Bayesian Optimization, are designed to balance exploration and exploitation to identify optimal configurations or policies dynamically. In the context of large-scale ranking, AE has been utilized to automate the tuning of system parameters and personalization strategies. For instance, recent works at Meta (Olson et al., 2025; Wu et al., 2022; Bakshy et al., 2018) demonstrate how AE can be deployed to efficiently search vast parameter spaces, allowing ranking systems to adapt to user feedback more rapidly than traditional A/B testing cycles would permit.

3 Preliminary

3.1 Notation

Experimental Setup and Notation We consider a randomized online experiment (A/B test) with a set of M actions (treatments)

$$\mathcal{A} = \{a_1, \ldots, a_M\},$$

and a control action a_0. We evaluate K performance metrics $\{\delta_1, \ldots, \delta_K\}$. Each user u_i is associated with a D-dimensional feature vector

$$\{X_1(u_i), \ldots, X_D(u_i)\}, \quad X_d(u_i) \in \mathcal{X}.$$

Users are randomly assigned to one of the treatment groups $\{U_1, \ldots, U_M\}$ or the control group U_0. We write $\delta_k(u_i, a_j)$ for the observed outcome of user u_i under action a_j on metric δ_k.

Average Treatment Effect (ATE) The average treatment effect of treatment a_j relative to control a_0 on metric δ_k is

$$\mathrm{ATE}(a_j, \delta_k) = \frac{1}{|U_j|} \sum_{u_i \in U_j} \delta_k(u_i, a_j) - \frac{1}{|U_0|} \sum_{u_i \in U_0} \delta_k(u_i, a_0).$$

Heterogeneous Treatment Effect (HTE) The individual-level heterogeneous treatment effect of treatment a_j relative to control a_0 for user u_i on metric δ_k is

$$\mathrm{HTE}(u_i, a_j, \delta_k) = \delta_k(u_i, a_j) - \delta_k(u_i, a_0).$$

By leveraging flexible architectures, HTE are able to model complex interactions between user features and interventions, and have seen widespread adoption in areas like online marketing and precision targeting.

3.2 GAS

GAS (Generalized Automatic Segmentation) (Wu et al., 2025) is a user-segment level HTE algorithm to improve personalization strategies and overcome obstacles. GEARS builds its candidate generation stage on top of GAS.

Let $\mathcal{S}$ denote a collection of non-overlapping user segments, and let $S \in \mathcal{S}$ be a segment (a subset of users). The segment-level heterogeneous treatment effect of a_j relative to a_0 on metric δ_k is

$$\mathrm{HTE}(S, a_j, \delta_k) = \frac{1}{|S|} \sum_{u_i \in S} \left(\delta_k(u_i, a_j) - \delta_k(u_i, a_0)\right).$$

Policy Parameterization through Quantile-Based Segmentation. GAS defines candidate policies through quantile-based cohort segmentation over user features. Let $Q(X, p)$ denote the $100p$-th percentile of feature $X \in \mathcal{X}$, with $Q(X, 0) = -\infty$. Let $N \in \mathbb{Z}_+$ denote the number of quantile bins.

Individual split. For feature X, define N segments:

$$S_i^{\text{ind}}(X) = \left\{ u \,\middle|\, Q\left(X, \frac{i-1}{N}\right) < X(u) \leq Q\left(X, \frac{i}{N}\right) \right\}.$$

Binary split. For threshold index $i_0 \in \{1, \ldots, N-1\}$:

$$S_{1,i_0}^{\text{bin}}(X) = \left\{ u \,\middle|\, Q(X, 0) < X(u) \leq Q\left(X, \frac{i_0}{N}\right) \right\},$$

$$S_{2,i_0}^{\text{bin}}(X) = \left\{ u \,\middle|\, Q\left(X, \frac{i_0}{N}\right) < X(u) \leq Q(X, 1) \right\}.$$

These segmentation strategies define a large combinatorial space of candidate policies across cohorts and treatments.

Finding Pareto Policies A policy is a mapping from segments to actions and can be represented as a set of (segment, action) pairs:

$$\mathcal{P} = \{(S_1, a_1), \ldots, (S_B, a_B)\},$$

where the segments $\{S_b\}_{b=1}^B$ form a partition of the user population.

In order to identify Pareto-optimal policies that jointly optimize multiple metrics, GAS adopts an efficient *weight-search* procedure. Specifically, it defines a family of *linear scalarized rewards* over K metrics using a set of weight vectors $\mathbf{W} \subset \mathbb{R}^K$. For each segment S_b (with associated action/policy a_b), GAS evaluates the weighted average metric lift $\delta_k(u_i, a_b)$ across users $u_i \in S_b$, and selects the set of policies induced by maximizing this scalarized objective over $\mathbf{w} \in \mathbf{W}$. The resulting Pareto policy set is

$$\mathcal{P}_{\text{pareto}} := \left\{ \arg\max_a \sum_{b=1}^{B} \sum_{k=1}^{K} \frac{w_k}{|S_b|} \sum_{u_i \in S_b} \delta_k(u_i, a_b) \,\middle|\, \mathbf{w} \in \mathbf{W} \right\}.$$

4 The GEARS System

Motivation. Existing personalization frameworks, including deep uplift models (Künzel et al., 2019; Wei et al., 2024) and generative recommendation approaches, treat ranking optimization as static signal estimation. These methods can identify statistically promising interventions but operate without access to *engineering context*: infrastructure constraints, feature-stability properties, and deployment criteria that determine whether a policy is feasible in production. For instance, an HTE model may surface a high-lift cohort defined by a feature that exhibits high temporal drift, yielding a policy that is optimal

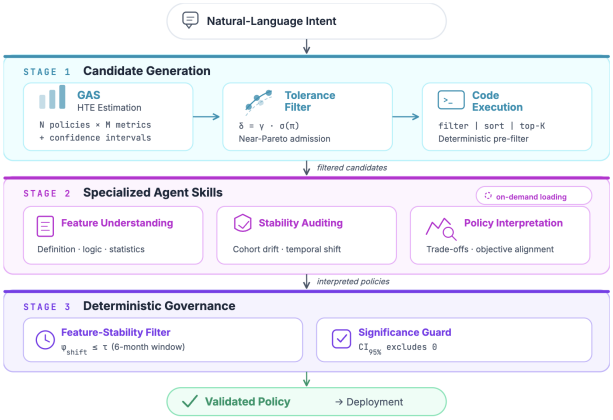

Figure 1 End-to-end GEARS pipeline. The system accepts a natural-language intent, generates and filters candidate policies through programmatic execution (Stage 1), interprets them using domain-specific skills loaded on demand (Stage 2), and validates production readiness through deterministic governance hooks (Stage 3).

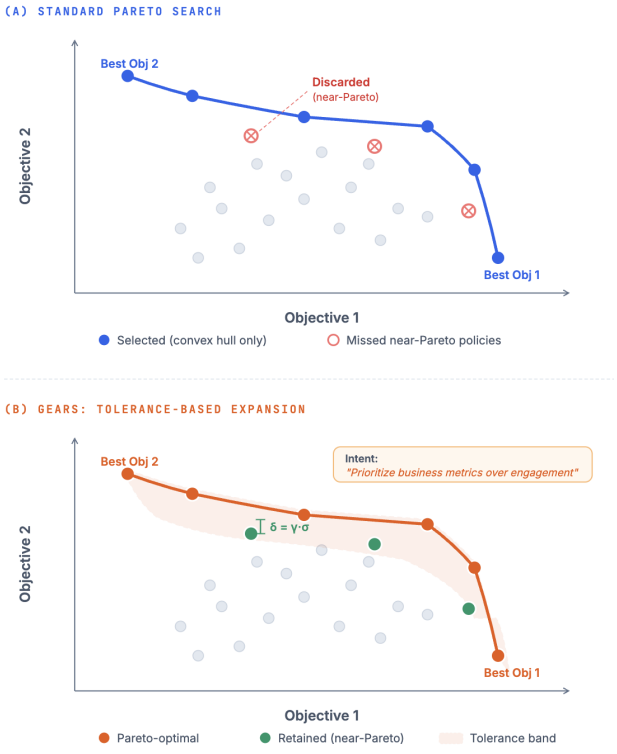

Figure 2 Tolerance-based frontier expansion. (a) Standard Pareto search selects only convex-hull solutions, discarding near-optimal candidates in non-convex regions. (b) GEARS admits policies within an uncertainty-aware tolerance band $\delta = \gamma \cdot \sigma$, retaining near-Pareto candidates that may offer better stability or deployability.

offline but undeployable. We refer to this disconnect between statistical optimality and production feasibility as the *deployment gap*.

GEARS bridges this gap by decomposing the end-to-end optimization workflow into three coordinated stages: programmatic filtering over the candidate space, domain-knowledge retrieval through specialized agent skills, and deterministic validation via lifecycle governance hooks. As illustrated in Figure 1, an LLM-based controller orchestrates these stages, accepting a natural-language product intent as input and producing a validated, deployment-ready policy configuration.

Intent-Conditioned Candidate Generation. The first stage translates a natural-language objective (e.g., *"maximize retention without degrading latency"*) into an executable search over the policy space.

- **Tolerance-Based Frontier Expansion.** GEARS builds on GAS (§ 3.2), which estimates heterogeneous treatment effects across quantile-based user segments and enumerates candidate policies via random-weight Pareto search. Standard random-weight search converges on the convex hull of the Pareto frontier, potentially discarding non-convex candidates that may be preferable for deployment. GEARS introduces a tolerance-based Pareto filter (Figure 2; pseudocode in Algorithm 1): for each metric m and candidate policy π, a tolerance margin $\delta_m(\pi) = \gamma \cdot \sigma_m(\pi)$ is defined, where $\sigma_m(\pi)$ is the estimated standard error and $\gamma \geq 0$ is a hyperparameter. A candidate is retained unless another candidate dominates it on every metric even after accounting for these margins. This expands the candidate set to include near-Pareto policies whose stability or interpretability may favor deployment.

- **Programmatic Pre-Filtering.** Before LLM-based reasoning, the agent translates the user's intent into deterministic operations over the candidate table by generating and executing shell commands. For example, given *"maximize Metric 1 without regressing Metric 2,"* the agent produces a command that retains rows where the lower confidence bound of Metric 2 is non-negative, then sorts by Metric 1 in descending order. This step is deliberately non-neural: structured filtering over tabular data is more reliably handled by programmatic execution than by text-based reasoning, and applying it early reduces the candidate set that the subsequent reasoning stage must process. As the ablation study in §5.2 confirms, this programmatic stage contributes substantially to overall performance on structured selection tasks, consistent with prior findings that code-based execution outperforms pure prompting for data-manipulation operations (Wang et al., 2024).

Domain-Grounded Interpretation via Specialized Agent Skills. While programmatic filtering handles well-

Algorithm 1: Tolerance-based Personalization: Random-weight Top-K + Tolerance Pareto Filtering

Input: Policy set $\mathcal{P}$; metrics $m = 1, \ldots, M$ (assume maximize); estimated means $\mu_m(p)$ and uncertainties $\sigma_m(p)$ for each $p \in \mathcal{P}$; number of random weights W; top-K per weight K; tolerance hyperparameter $\tau \geq 0$.

Output: Final candidate policy set $\mathcal{C}_\tau$ (Pareto-optimal and near-Pareto policies).

Step 1: Candidate collection via random weight search.

Sample W weight vectors $\{w^{(i)}\}_{i=1}^{W}$ from the simplex Δ^{M-1};

Initialize candidate set $\mathcal{C} \leftarrow \emptyset$;

for $i \leftarrow 1$ **to** W **do**

  foreach $p \in \mathcal{P}$ **do**

    $S_{w^{(i)}}(p) \leftarrow \sum_{m=1}^{M} w_m^{(i)} \mu_m(p)$;

  Let $\mathcal{T}^{(i)} \leftarrow \textsc{TopK}\big(\{S_{w^{(i)}}(p)\}_{p \in \mathcal{P}}, K\big)$;

  // Top-K policies by weighted score

  $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{T}^{(i)}$;

Step 2: Tolerance-based Pareto filtering (near-frontier admission).

Define per-metric tolerance margin for candidate p:

$$\epsilon_m(p) \triangleq \tau \cdot \sigma_m(p).$$

Define *tolerance-dominance* $q \succ_\tau p$ (maximize case) as:

$$\Big(\forall m,\ \mu_m(q) \geq \mu_m(p) - \epsilon_m(p)\Big) \ \wedge$$

$$\Big(\exists m,\ \mu_m(q) > \mu_m(p) + \epsilon_m(p)\Big).$$

Initialize $\mathcal{C}_\tau \leftarrow \emptyset$;

foreach $p \in \mathcal{C}$ **do**

  dominated $\leftarrow$ **false**;

  foreach $q \in \mathcal{C}$ **do**

    if $q \neq p$ **and** $q \succ_\tau p$ **then**

      dominated $\leftarrow$ **true**;

      break;

  if not *dominated* **then**

    $\mathcal{C}_\tau \leftarrow \mathcal{C}_\tau \cup \{p\}$;

return $\mathcal{C}_\tau$;

defined, structured queries effectively, many practical decisions require contextual judgment that cannot be reduced to deterministic rules. For instance, determining whether a feature is suitable for long-term targeting, or explaining why a particular cohort split yields a favorable trade-off, requires domain-specific knowledge absent from both the LLM's pretraining data and the tabular experiment records.

GEARS encapsulates this knowledge in *Specialized Agent Skills*: modular, read-only resources that provide the agent with structured procedures and access to internal tools. Each skill consists of (i) a *metadata header* describing its purpose and trigger conditions, and (ii) a *core body* containing domain logic and tool-invocation templates. At inference time, only skill metadata is loaded into the agent's context; the core body is loaded on demand when the agent determines relevance, bounding context length and mitigating context rot in long sessions.

Three categories of skills are deployed:

- **Feature understanding** – retrieves the definition, computation logic, and historical statistics of a ranking feature, enabling assessment of whether a policy's reliance on a given feature is operationally sound.
- **Stability auditing** – queries internal monitoring tools to compute the user-cohort shift ratio (§5.3) for each feature in a candidate policy, flagging features with high temporal or distributional drift.
- **Policy interpretation** – summarizes the trade-off profile of a recommended policy and generates a natural-language rationale for human review before deployment.

The contribution of skills is complementary to programmatic filtering: they improve performance on tasks that require contextual interpretation beyond structured data manipulation.

Deterministic Lifecycle Governance. The final stage ensures that policies surfaced by the preceding components are robust enough for production deployment. GEARS registers *validation hooks*—deterministic checks executed before a candidate is promoted to the recommendation set.

Two hooks are enforced by default:

- **Feature-stability filter.** For every feature in a candidate policy, the hook computes the *user-cohort shift ratio* ϕ_{shift}: the fraction of users who migrate across cohort boundaries over a six-month window. A policy is admitted only if all its features satisfy $\phi_{\text{shift}} \leq \tau$, where τ is calibrated against an empirically stable baseline feature set (details in §**??**).
- **Statistical-significance guard.** The hook verifies that reported metric lifts are statistically significant (95% confidence interval excludes zero for the primary metric), filtering candidates whose gains may be attributable to sampling noise.

Because all hooks are deterministic and logged, they produce an auditable record of admission and rejection decisions, supporting both automated governance and human oversight.

4.1 Illustrative Agent Session

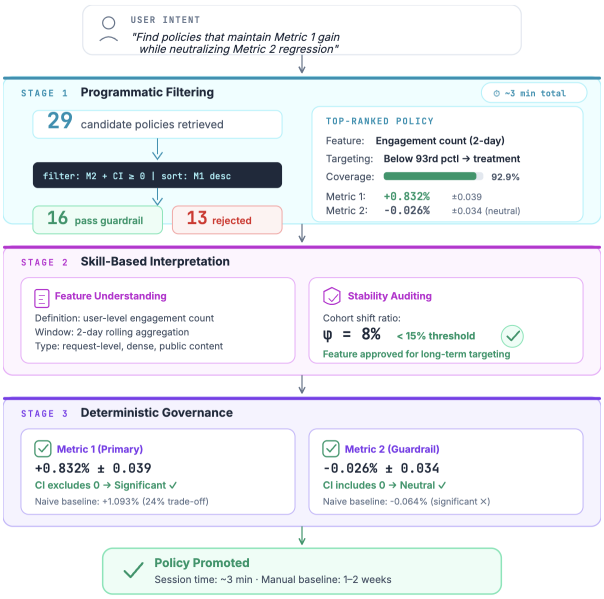

Figure 3 Anonymized trace of a production GEARS session. The system filters 29 candidates to 16 via programmatic guardrail checks (Stage 1), verifies feature stability through domain-specific skills (Stage 2), and validates statistical significance before promoting the policy (Stage 3). The targeted policy neutralizes the guardrail regression present in the naive baseline.

Figure 3 presents an anonymized trace of a production GEARS session. Given the intent *"maintain Metric 1 gain while neutralizing Metric 2 regression,"* the pipeline proceeds as follows. In Stage 1, GAS generates hundreds of candidate policies from a billion-user experiment; the tolerance-based Pareto filter retains 29 near-optimal candidates, and programmatic code execution applies the guardrail constraint (Metric 2 value + CI ≥ 0), reducing the set to 16. In Stage 2, the agent invokes the *feature understanding* skill to retrieve the definition of the top policy's segmentation feature (a two-day engagement count) and the *stability auditing* skill to verify its cohort shift ratio ($\phi_{\text{shift}} = 8\%$, below the 15% threshold). In Stage 3, governance hooks confirm that Metric 1 is statistically significant ($+0.832\% \pm 0.039$) and Metric 2 is neutral ($-0.026\% \pm 0.034$, CI includes zero). The resulting policy retains 76% of the naive baseline's Metric 1 gain while fully eliminating the guardrail violation.

5 Experiments

In this section, we investigate the following research questions:

- **RQ1**: How does each component of the GEARS pipeline contribute to policy selection performance?
- **RQ2**: Does the governance layer improve the deployability of recommended policies?
- **RQ3**: Can GEARS deliver measurable metric improvements in production deployment?

5.1 Experimental Settings

Dataset We constructed a benchmark dataset from 20 internal large-scale online experiments deployed on a live production system serving billions of users, with approximately 0.5 billion users per experiment and 10 billion user-experiment pairs in total (with overlap across experiments). For each experiment, we initially ran the GAS (Wu et al., 2025) algorithm to generate hundreds of policy candidates with their corresponding metric measurements. To enable objective evaluation with deterministic ground truth, we defined five instruction templates representing common policy selection scenarios:

- **Maximize Both**: Find policies that jointly optimize two metrics
- **Maximize with Constraint**: Optimize a primary metric while ensuring a secondary metric does not regress
- **Tradeoff Analysis**: Identify Pareto-optimal policies representing different tradeoff points between competing metrics
- **Efficiency Optimization**: Select policies with the highest composite efficiency score
- **Single Metric**: Maximize a single target metric regardless of others

Instantiating each template across 20 experiments yields 100 structured instructions. For each, the ground-truth ranking is computed deterministically from the optimization criteria, enabling unambiguous evaluation.

These structured instructions are well suited to programmatic execution; the purpose of this benchmark is to measure the reliability of the full pipeline on tasks with verifiable answers, not to evaluate open-ended reasoning.

Baselines To assess the effectiveness of our proposed GEARS framework, we benchmark its performance against several established prompting strategies:

- **Naive Prompting**: Directly queries the LLM with the task instruction and data without additional reasoning guidance.
- **Chain-of-Thought (CoT)** (Wei et al., 2022): Encourages step-by-step reasoning by prompting the model to first understand the objective, analyze the data, apply selection criteria, and then provide recommendations.
- **Self-Consistency** (Wang et al., 2022): Samples multiple reasoning paths with temperature-based diversity ($T = 0.7$) and aggregates predictions via Borda count voting to improve robustness.
- **Self-Refine** (Madaan et al., 2023): A two-stage approach where the model first generates initial recommendations, then critically reviews and refines its own output to correct potential errors.
- **Code-as-Action** (Wang et al., 2024): Instead of only generating text, the LLM generates and executes code to solve the task. This makes outputs verifiable and reproducible, reducing errors and hallucinations in computation- or data-driven settings.

Metrics To evaluate policy selection performance, we employ standard ranking and retrieval metrics widely adopted in information retrieval and recommender systems.

- **Precision@K** measures the fraction of recommended policies within the top-K that are included in the ground-truth set.
- **Recall@K** quantifies the proportion of ground-truth policies that appear within the top-K predictions.
- **NDCG@K** (Normalized Discounted Cumulative Gain) evaluates not only whether the model retrieves relevant policies but also whether they are ranked near the top:

$$\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}, \text{DCG@K} = \sum_{i=1}^{K} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)},$$

  where $\text{rel}_i \in \{0, 1\}$ indicates whether the policy at position i belongs to the ground-truth set.
- **Top-1 Accuracy** measures whether the highest-ranked prediction exactly matches the best ground-truth policy.
- **Top-1 in GT** reports whether the top-ranked prediction belongs to the ground-truth set (a relaxed version of Top-1 Accuracy).
- **Ranking Correlation** assesses the agreement between predicted and ground-truth rankings via Spearman's ρ, capturing global ordering fidelity.

We report results for $K \in \{1, 3, 5\}$. Together, these metrics capture complementary aspects of performance: correctness (Top-1 Accuracy), coverage (Recall@K), precision-coverage tradeoff (Precision@K), ranking quality (NDCG@K), and global ordering (Ranking Correlation).

Implementation Details All experiments use Claude Sonnet 4.6 (Anthropic, 2025c) as the backbone LLM. For Self-Consistency, we sample 5 responses per instruction with temperature 0.7 and aggregate via Borda count. For Self-Refine, we use a single refinement iteration.

Table 1 Quantitative comparison of policy selection performance. We evaluate the proposed GEARS framework against five state-of-the-art baselines. Performance is measured across five key dimensions: Ranking Quality (nDCG@k), Precision (Prec@k), Global Rank Correlation, Recall (Rec@k), and Top-1 Performance.

Method	Ranking Quality			Precision@k			Global	Recall@k			Top-1 Performance	
	nDCG@1	nDCG@3	nDCG@5	Prec@1	Prec@3	Prec@5	Rank Corr.	Rec@1	Rec@3	Rec@5	Top-1 Acc	Top-1 in GT
Naive	0.57	0.70	0.74	0.57	0.39	0.28	0.20	0.36	0.67	0.77	0.44	0.57
CoT Wei et al. (2022)	0.68	0.80	0.83	0.68	0.44	0.30	0.31	0.43	0.73	0.82	0.57	0.68
Self-Consistency Wang et al. (2022)	0.57	0.74	0.76	0.57	0.39	0.28	0.13	0.33	0.67	0.78	0.37	0.57
Self-Refine Madaan et al. (2023)	0.61	0.78	0.80	0.61	0.44	0.31	0.34	0.37	0.74	0.83	0.50	0.61
Code-as-Action Wang et al. (2024)	0.77	0.87	0.87	0.77	0.52	0.33	0.59	0.45	0.84	0.88	0.68	0.77
GEARS w/o Bash	0.40	0.42	0.42	0.40	0.18	0.11	0.80	0.24	0.32	0.32	0.26	0.40
GEARS w/o Skill	0.87	0.91	0.91	0.87	0.57	0.35	0.72	0.53	0.89	0.90	0.77	0.87
Ours (GEARS)	**0.94**	**0.96**	**0.96**	**0.94**	**0.60**	**0.37**	**0.82**	**0.56**	**0.94**	**0.95**	**0.86**	**0.94**

5.2 Structured Policy Selection and Component Analysis

We evaluate GEARS under an *offline policy selection* setting, where the model is given tabular experiment records of multiple candidate policies and is instructed to output a ranked list of recommended policies. Each candidate policy is associated with multiple metrics (e.g., primary objective and guardrail metrics).

Table 1 reports the policy selection performance across a suite of ranking and decision-oriented metrics. GEARS consistently outperforms all baselines across most metrics, indicating stronger reliability in selecting high-quality policies under multi-metric constraints.

We further conduct ablation studies to isolate the contribution of each component in GEARS. On structured instructions, removing code execution (GEARS w/o Bash) causes the largest performance drop (nDCG@1: 0.94 $\rightarrow$ 0.40), confirming that programmatic filtering is the primary driver for deterministic selection tasks—an expected result, since these instructions map directly to tabular operations (filter, sort, rank) that are more reliably executed as code than reasoned about in text.

Removing skills (GEARS w/o Skill) yields a smaller but consistent drop (nDCG@1: 0.94 $\rightarrow$ 0.87). On structured tasks, skills provide incremental gains by improving the agent's interpretation of domain-specific selection criteria. In production, where instructions are less structured and feature context is critical, the contribution of skills is substantially larger.

5.3 Hooks Improve the Reliability of GEARS Recommendation

To ensure that the policies generated by GEARS are deployable and robust against temporal distribution shifts, we established a rigorous quantitative benchmark for checking *feature stability*.

5.3.1 Benchmarking Methodology

We define the *User-Cohort Shift Ratio* (R_{shift}) as the primary metric for stability: the percentage of users who migrate from their assigned cohort (bucket) to a different one over a 6-month window. To set a realistic baseline, we benchmarked a *perceived stable* feature set $\mathcal{S}$ using two cohort definitions: Quantile Cuts (4 equal buckets) and Binary Cuts (p25/p75 thresholds).

Table 2 Feature Stability Benchmark (6-Month Window). Binary cuts align engagement feature stability with feature set $\mathcal{S}$ baselines.

Feature Class	Feature Name	Shift (Quantile)	Shift (Binary)	Status
Baseline	Feature Set $\mathcal{S}$	6%	2%	Benchmark
Product	Feature 2	16%	4%	Stable
Engagement	Feature 3	30%	10-12%	Stable (Binary)
Product	Feature 4	~50%	~20%	Unstable
Product	Feature 5	N/A	~30%	Unstable

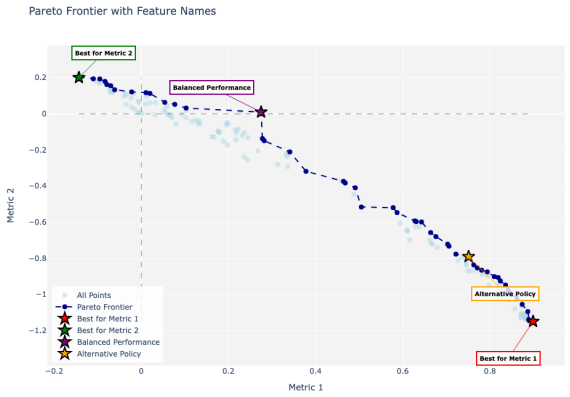

Figure 4 Pareto efficiency of generated policies. We plot the performance of all candidate policies, with the Pareto frontier (dark blue line) indicating the optimal trade-off curve. Annotated stars mark key policies of interest.

5.3.2 Empirical Baselines and Thresholds

Our analysis revealed that even the baseline exhibits drift. As shown in Table 2, feature set $\mathcal{S}$ shifted by 6% over 6 months, establishing a lower bound for unavoidable natural drift. Product features, such as *Feature 4*, showed high volatility (50% shift) under quantile cuts, making

them unsuitable for long-term policy targeting.

Based on these benchmarks, we implemented a validation hook within GEARS:

- **Pre-Search Filter:** Features must exhibit $R_{\text{shift}} \leq 15\%$ (Binary) or $\leq 45\%$ (Quantile) to enter the search space.

This benchmarking process allowed GEARS to automatically disqualify high-lift but unstable features (e.g., Feature 4) that baseline methods would have erroneously selected.

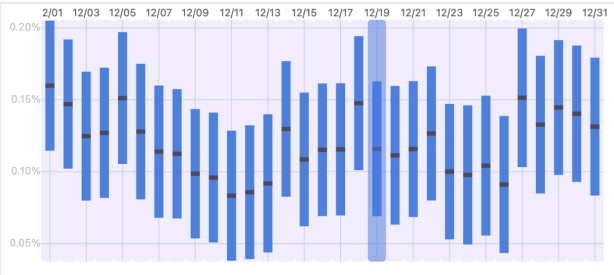

Figure 5 The backtest results indicate that the metrics improvement achieved by the selected policy remains consistent over a period of one month.

As illustrated in Figure 4, to validate the efficacy of our stability governance, we conducted a controlled selection from a randomized experiment with four policy candidates. After filtering out high-variance candidates, the remaining policy (*Best for Metric 2*) demonstrated consistent metric improvement over a one-month period Figure 5.

5.4 Broad Adoption and Real-World Impact

Table 3 Experimental results of GEARS across diverse surfaces. A dash (—) indicates that the specific metric was not applicable or not the primary optimization target for that surface.

Domain	Metric 1 (%)	Metric 2 (%)	Metric 3 (%)
Surface 1	0.14	—	—
Surface 2	—	—	0.08
Surface 3	0.10	0.089	—
Surface 4	0.10	—	—
Surface 5	0.042	—	—
Surface 6	0.011	0.017	0.08
Surface 7	0.0406	—	—
Surface 8	0.13	0.37	—
Surface 9	—	0.044	0.02

Table 3 demonstrates that GEARS delivers improvements across various experimental surfaces. In each deployment, we observe clear and measurable gains in the key metrics. Taken together, these results support the paper's claim that GEARS is an effective, general-purpose mechanism for turning cohort-level personalization into repeatable efficient general Agent framework, rather than a one-off optimization tied to a single surface or a single metric.

5.5 Cost Analysis

GEARS operates entirely offline and does not affect user-facing latency. A typical session consumes approximately 3.9K input tokens and 35.8K output tokens, costing roughly $3.75 per run. At a scale of 400 experiments per year, the annual LLM cost is approximately $1,500. Prior to GEARS, the equivalent manual workflow (launching HTE flows, querying feature definitions, evaluating feasibility) required 1–2 weeks of engineering time; GEARS completes this pipeline in 1–2 days.

6 Practical Evaluation

We present an anonymized case study to illustrate how GEARS can recommend actionable optimization policies under multi-objective constraints with minimal manual iteration. In § 6.1, we study a large-scale recommendation setting where two primary objectives exhibit a consistent trade-off, and show how GEARS discovers cohorts that admit differentiated treatments while respecting additional guardrail metrics.

6.1 Complex Trade-off Optimization in Large-Scale Recommendation

Table 4 Quantitative results for Baseline Treatment Arms. The table reports the percentage lift relative to the control group (mean ± standard error).

Treatment Arm	Metrics 1	Metrics 2
Treatment 1	$-0.049\% \pm 0.043$ ✗	**+0.282% ± 0.074** ✓
Treatment 2	**+0.036% ± 0.034** ✓	$-0.289\% \pm 0.073$ ✗

In large-scale recommendation systems, improving one engagement objective often comes at the expense of another, creating a persistent multi-objective optimization challenge where globally uniform treatments can lead to near zero-sum outcomes.

As shown in Table 4, two competing global variants highlight this tension: a treatment improves Metric 1 but degrades Metric 2, while another treatment improves Metric 2 at the cost of Metric 1.

GEARS addresses this by replacing manual slice-and-dice analyses with an agentic personalization workflow. Given a high-level natural language prompt (e.g., *How can I find the tradeoff between metric 1 and metric 2 for this experiment*), the agent iteratively explores a high-dimensional cohort space, proposes candidate segments, and validates them against both primary objectives and

guardrail metrics. The outcome is a set of cohort-specific policies where at least one primary objective improves with statistical significance while the other objectives remain neutral within acceptable bounds. Table 5 reports the top-3 policies recommended by GEARS. All three achieve statistically significant gains on Metric 2 while maintaining neutrality on Metric 1.

GEARS has found that different types of users react differently to changes in the mix of content they see. For example, users who are very active benefit more from the first treatment, which improves one key metric but doesn't affect another much. On the other hand, less active users prefer the second treatment, which helps them become more engaged.

Table 5 Top-3 policies recommended by GEARS for the case study. All three achieve statistically significant gains on Metric 2 while maintaining neutrality on Metric 1 (95% CI includes zero).

	Metric 1		**Metric 2**	
Policy	Lift (%)	95% CI	Lift (%)	95% CI
1	−0.032	±0.057	**+0.350**	±0.092
2	−0.029	±0.057	**+0.349**	±0.092
3	−0.025	±0.057	**+0.338**	±0.092

Deploying the resulting cohort-targeted policy yields a statistically significant lift on the prioritized metric while maintaining neutrality on the competing metric. Operationally, GEARS automated what was previously a multi-week, expert-driven discovery process, and leverages the multi-agent system to explain how the decision was made and the rationale behind the decision in current ranking context. This significantly improves system efficiency and enabling broader exploration of the policy space through effective human-AI collaboration.

References

Xavier Amatriain. Prompt design and engineering: Introduction and advanced methods. *arXiv preprint arXiv:2401.14423*, 2024.

Anthropic. Model context protocol, 2024. https://www.anthropic.com/news/model-context-protocol. Accessed: 2026-01-12.

Anthropic. Hooks guide, 2025a. https://code.claude.com/docs/en/hooks-guide. Accessed: 2026-01-12.

Anthropic. Agents and tools: Agent skills overview, 2025b. https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview. Accessed: 2026-01-12.

Anthropic. Introducing claude 4, 2025c. https://www.anthropic.com/news/claude-4. Accessed: 2026-01-12.

Susan Athey and Guido Imbens. Recursive partitioning for heterogeneous causal effects. *Proceedings of the National Academy of Sciences*, 113(27):7353–7360, 2016.

Eytan Bakshy, Lili Dworkin, Brian Karrer, Konstantin Kashin, Benjamin Letham, Ashwin Murthy, and Shaun Singh. Ae: A domain-agnostic platform for adaptive experimentation. In *Conference on neural information processing systems*, pages 1–8, 2018.

Ioana Bica, James Jordon, and Mihaela van der Schaar. Estimating the effects of continuous-valued interventions using generative adversarial networks. *Advances in Neural Information Processing Systems*, 33:16434–16445, 2020.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2022.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. arXiv preprint, 2025.

Sören R Künzel, Jasjeet S Sekhon, Peter J Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the national academy of sciences*, 116(10):4156–4165, 2019.

Akos Lada, Alexander Peysakhovich, Diego Aparicio, and Michael Bailey. Observational data for heterogeneous treatment effects with application to recommender systems. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 199–213, 2019.

Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Fei-Fei Li, Fei Xia, and Brian Ichter. Chain of code: Reasoning with a language model-augmented code emulator. *International Conference on Machine Learning*, 2023.

Christos Louizos, Uri Shalit, Joris M Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. *Advances in neural information processing systems*, 30, 2017.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, S. Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, A. Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *Neural Information Processing Systems*, 2023.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, et al. A survey of context engineering for large language models. *arXiv preprint arXiv:2507.13334*, 2025.

G. Mialon, Roberto Dessì, M. Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, R. Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey. *Trans. Mach. Learn. Res.*, 2023.

Preetam Nandy, Xiufan Yu, Wanjun Liu, Ye Tu, Kinjal Basu, and Shaunak Chatterjee. Generalized causal tree for uplift modeling. In *2023 IEEE International Conference on Big Data (BigData)*, pages 788–798. IEEE, 2023.

Miles Olson, Elizabeth Santorella, Louis C Tiao, Sait Cakmak, Mia Garrard, Samuel Daulton, Zhiyuan Jerry Lin, Sebastian Ament, Bernard Beckerman, Eric Onofrey, et al. Ax: A platform for adaptive experimentation. In *AutoML 2025 ABCD Track*, 2025.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph Gonzalez. Memgpt: Towards llms as operating systems, 2023. https://arxiv.org/abs/2310.08560v2.

Alexander Peysakhovich and Akos Lada. Combining observational and experimental data to find heterogeneous treatment effects. *arXiv preprint arXiv:1611.02385*, 2016.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs, 2025. https://arxiv.org/abs/2504.13958v1.

Nicholas J Radcliffe and Patrick D Surry. Real-world uplift modelling with significance-based uplift trees. *White Paper TR-2011-1, Stochastic Solutions*, pages 1–33, 2011.

Claudia Shi, David Blei, and Victor Veitch. Adapting neural networks for the estimation of treatment effects.

Advances in neural information processing systems, 32, 2019.

Noah Shinn, Federico Cassano, Beck Labash, A. Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. *Neural Information Processing Systems*, 2023.

Zexu Sun and Xu Chen. M 3 tn: Multi-gate mixture-of-experts based multi-valued treatment network for uplift modeling. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5065–5069. IEEE, 2024.

J. D. Velásquez-Henao, Carlos Jaime Franco-Cardona, and Lorena Cadavid-Higuita. Prompt engineering: a methodology for optimizing interactions with ai-language models in the field of engineering. *DYNA*, 2023.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yuxiang Wei, Zhaoxin Qiu, Yingjie Li, Yuke Sun, and Xiaoling Li. Multi-treatment multi-task uplift modeling for enhancing user growth. *arXiv preprint arXiv:2408.12803*, 2024.

Han Wu, Sarah Tan, Weiwei Li, Mia Garrard, Adam Obeng, Drew Dimmery, Shaun Singh, Hanson Wang, Daniel Jiang, and Eytan Bakshy. Interpretable personalized experimentation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4173–4183, 2022.

Yihan Wu, Mingze Gao, Haoran Liu, Weiwei Li, Kevin Han, Junfeng Pan, Xinyi Zhang, Jiawei Wen, and Gedi Zhou. Gas: Large-scale heterogeneous personalization in social network applications at meta. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, page 5049–5058, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714542. doi: 10.1145/3711896.3737225. https://doi.org/10.1145/3711896.3737225.

Qinyuan Ye, Mohamed Ahmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 355–385, 2024.

Yan Zhao, Xiao Fang, and David Simchi-Levi. Uplift modeling with multiple treatments and general response types. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 588–596. SIAM, 2017.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory, 2023. https://arxiv.org/abs/2305.10250.